平面グラフの2連結化アルゴリズム (グラフ理論とその応用)

AUTHOR(S):

小野口, 一則; 千葉, 則茂; 西関, 隆夫

数理解析研究所講究録
534巻 1984年 221-233

平面グラフの2連結化アルゴリズム

東北大工学部　小野口一則 (Kazunori Onoguchi)
千葉　則茂 (Norishige Chiba)
西関　隆夫 (Takao Nishizeki)

1　まえがき

与えられたグラフGに枝を何本か付加することにより2連結グラフにすることを2連結化といい，通信網の信頼性の向上などに応用される．例えば，通信網の中継局を点に伝送路を枝に対応させたグラフを2連結化することは，中継局のうちどの1箇所が故障しても通信が途絶えることのない通信網に拡張することに対応する．グラフを2連結化する場合，付加しなければならない枝の最小本数が何本かということが問題となる．Eswaran とTarjan[1] は一般グラフを2連結化するのに必要な枝の最小本数に関する必要十分条件を与えておりEswaran とTarjan[1] ，Rosenthal とGoldner[2] はその枝を具体的に求める線形時間アルゴリズムが得られたと主張している．しかしこの2つの文献のアルゴリズムには少し誤りがあり，そのままでは最小枝数で2連結化するとは限らない．本文では，まず実際に最小本数の枝付加で一般グラフを2連結化する線形時間アルゴリズムを与える．この場合，平面グラフが与えられても枝付加により平面性が損われるかもしれない．次に，対象とするグラフを平面グラフに限定し，その平面埋め込みを固定したまま最小本数の枝を付加して2連結平面化する線形時間アルゴリズムを与える．この場合には，グラフの平面性が損われるような枝付加はできず，従来のアルゴリズムは適用できない．

2　準備

ここでは本文で用いる用語を定義する．切断点を含まないGのの2連結成分を孤立ブロックと呼ぶ．G自身ではないGの孤立ブロックはq(≥ 0)個あるとする．切断点を丁度1個だけ含むGの2連結成分をペンダントブロックと呼ぶ．Gにはペンダントブロックがp個あるとする．vをGの任意の点とする．Gの枝は次の同値類E_iに分割できる．

2本の枝 (x, y) と (w, z) が同じ同値類に含まれるのは，それらを含む道 (v_1, v_2)，(v_2, v_3)，$\cdots$，(v_{k-1}, v_k) が存在し，しかも$2 \leq i \leq k-1$なるiに対し$v_i \neq v$である時であり，かつその時に限る．

この同値類E_iを用いてGの部分グラフG_iを次のように定義する．

$G_i = (V_i, E_i)$
$V_i = \{x \in V \mid (x, y) \in E_i\}$

この部分グラフG_iをGのv-ブロックと呼び，Gのv-ブロックの個数を$d(v)$と書く．dを$d = \max\{d(v) \mid v \in V\}$と定義する．Gの切断点及び2連結成分を点で表わし，G上で隣接している切断点と2連結成分を枝で結んだ木をblock-cutpoint木（BC木）と呼び，T(G)と書く．T(G)上で切断点に対応する点をC点，2連結成分に対応する点をB点と呼び，図上ではC点は黒丸で，B点は白丸で表わす．Gのペンダントブロックは T(G)の葉に対応する．T(G)上でC点vから葉ℓまでの道をQ＝(v, v_1)，(v_1, v_2)，$\cdots$，(v_m, ℓ)としよう．枝(v_1, v_2)をQの始点の枝，(v_m, ℓ)を終点の枝と呼ぶ．Gは2連結でないので，面の境界は必ずしも単純な閉路ではないが，面の境界を面閉路と呼ぶことにする．Gが平面グラフの時，ある1つの面F_iに着目する．F_iの面閉路上の点及び枝のみからなるGの部分グラフG_iをF_iの面グラフと呼ぶ．面F_iに含まれるペンダントブロックP_iとはP_iの外周上のすべての枝がF_iの面閉路上にあるペンダントブロックのことをいう．図1で面F_1に含まれるペンダントブロックはP_1およびP_2であり，面F_2に含まれるペンダントブロックはP_3，P_4およびP_5である．また隣合ったペンダントブロックとはF_iの面閉路を時計回りにたどった時に隣り合って探索されるペンダントブロックのことをいう．図1でP_3とP_4，P_4とP_5，P_5とP_3，P_1とP_2はそれぞれ隣り合ったペンダントブロックである．

3　一般グラフの2連結化アルゴリズム

3．1　アルゴリズム

一般グラフGに枝を付加して2連結化するのに必要な枝の本数の最小値を$\mu(G)$と書くことにする．次の補題がEswaran とTarjan[1] によって示されている．

［補題1］$\mu(G) = \max\{d-1, \lceil p/2 \rceil + q\}$　　（補題終）

本章では，一般のグラフGが与えられた時，Gに$\mu(G)$本の枝を付加して2連結化する線形時間アルゴリズムBICONNECT を与える．この場合，たとえGが平面グラフであっても枝付加によりグラフが非平面になることもある．

このアルゴリズムは補題1の$\mu(G) \leq \max\{d-1, \lceil p/2 \rceil + q\}$の証明に基ずいているため，この証明を示しておく．

（補題1の証明）

$i = \max\{d-1, \lceil p/2 \rceil + q\}$に関する帰納法で証明する．Gが2連結グラフでない場合を考えるので$i \geq 1$として一般性を失わない．

（1）$i = 1$及び2の時

$i = 1$となり得るのは，$d = p = 2$，$q = 0$の場合のみである．この場合，2つのペンダントブロックを切断点以外で1本の枝で結ぶとGは2連結グラフになる．したがって$\mu(G)=1$である．$i = 2$となり得るのは，$d = 2$，3及び$p = 3$，4の場合のみである．$p = 4$の場合で，ペンダントブロックを丁度2つ含むv-ブロックがある時はそのペンダントブロックの1つと，他のv-ブロック中のペンダントブロックとを切断点以外で1本の枝で結ぶ．それ以外の場合は，かってな2つのペンダントブロックを切断点以外で1本の枝で結ぶ．するといずれの場合も$d = p = 2$となり明らかに$\mu(G)=2$である．

（2）$i > 2$の時

$\max\{d-1, \lceil p/2 \rceil + q\} = i$のとき，この補題が成り立つと仮定する．いま$\max\{d-1, \lceil p/2 \rceil + q\} = i + 1$とし，いくつかの場合に分けて考える．

(2-1) Gが非連結の時

2つの連結成分G_1，G_2に着目する．G_1，G_2は孤立ブロックであるか，さもなければペンダントブロックを含んでいる．G_1とG_2を1本の枝で結ぶ．ただしG_1が孤立ブロックでないならば，その枝の1つの端点はG_1のペンダントブロック内の切断点でない点とする．その付加する枝のもう1つの端点をG_2内に同様に選ぶとする．得られるグラフにおけるp,d,q は次のようになり$\max\{d-1, \lceil p/2 \rceil + q\} = i$を満たすことが解る．

（Ⅰ）G_1及びG_2がともに孤立ブロックの場合

$p := p + 2$，$q := q - 2$，$d := d - 1$

（Ⅱ）G_1あるいはG_2の片方だけが孤立ブロックの場合

p変化なし，$q := q - 1$，$d := d - 1$

（Ⅲ）G_1，G_2とも孤立ブロックでない場合

$p := p - 2$，q変化なし，$d := d - 1$

仮定より，このグラフはi本の枝付加で2連結化できるので，結局Gはi＋1本の枝付加で2連結化できる．

(2-2) Gが連結の時

このときq＝0である．次の3つの場合がある．

(2-2-1) $d - 1 > \lceil p/2 \rceil$の時

このとき$d(v)=d$である切断点vは1個しかない（2つ以上あるとすると$d - 1 \leq \lceil p/2 \rceil$となり仮定に反する）．この切断点vのv-ブロックのうち，異なる2つのv-ブロックからそれぞれ1つずつペンダントブロックを選び，切断点以外で1本の枝で結ぶ．すると2つのv-ブロックが1つになるためdが1つだけ減少し，新しいグラフG'は$\max\{d'-1, \lceil p'/2 \rceil\} = i$を満たす．ここで，d'，p' はグラフG'におけるd，pである．仮定より，G'はi本の枝付加で2連結化できるのでGはi＋1本の枝付加で2連結化できる．

(2-2-2) $d - 1 = \lceil p/2 \rceil$の時

このときd(v)=dである切断点vは高々2つしかない．このような切断点vのv-ブロックの中で，2つ以上ペンダントブロックを持つv-ブロック（それをG_1とする）を選ぶ．もしd(w)=dを満たすv以外の切断点wが存在するなら，その点wはG_1に含まれており，点vのG_1以外の各v-ブロックにはペンダントブロックは1個しか含まれていない．G_1に含まれているペンダントブロックと他のv-ブロックに含まれているペンダントブロックを1つずつ選び，切断点以外で1本の枝で結ぶ．するとpが少なくとも2減少し，かつ2つのv-ブロックが1つになるためdも1つ減少する．d(w)=dなる切断点がもし存在するならそれはG_1中に存在するため，w-ブロックの数も1つだけ減少する．よって，得られたグラフG'はmax{d'-1, ⌈p'/2⌉} = iを満たす．仮定より，G'はi本の枝付加で2連結化できるのでGはi+1本の枝付加で2連結化できる．

(2-2-3) d−1<⌈p／2⌉の時

d(v)=dである任意の切断点vに注目する．ただし，d=2かつp>3の場合次のようにvを選び直す．（この部分が文献(1) および(2) で欠けている．下の注意参照）v以外の切断点v_0があって，そのv_0−ブロックBでちょうど2つだけペンダントブロックを含むものにvが含まれるならv_0を新たにvとして選ぶ．例えば図2(a)において最初に注目した切断点vが点eまたはfであるなら点aをvとして選び直す．

いまv-ブロックの中には2つ以上ペンダントブロックを含むものが必ず存在するため，そのようなv-ブロックに含まれているペンダントブロックとそれ以外のv-ブロックに含まれているペンダントブロックを1つずつ選び，それらを切断点以外で1本の枝で結ぶ．すると⌈p/2⌉が1つ減少するため得られたグラフG'はmax{d'-1, ⌈p'/2⌉}を満たす．仮定より，G'はi本の枝付加で2連結化できるのでGはi+1本の枝付加で2連結化できる．

（証明終）

［注意］

d=2かつp>3の場合，すべての切断点vがd(v)=d=2を満たすが，上述のように切断点vの選び方に制約を加えている．これは1本枝を付加することにより新しいペンダントブロックが出現する場合があるからである．例えば，図2(a) のグラフの場合，d=2，p=6であるが切断点e（d(e)=2）に注目すると，ペンダントブロック1及び5が異なるe-ブロック（一方は2つ以上ペンダントブロックを持つ）に含まれているため，この2つが枝で結ばれる可能性がある．このとき，図2(b) のようにaを切断点とするペンダントブロックが新たに1つ出現するため，全体としてペンダントブロックの個数は1つしか減らず，帰納法がうまく働かない．このように新しいペンダントブロックができるのは，ちょうど2つのペンダントブロックを含んでいるv-ブロック（図2のa-ブロック）が存在し，そのv-ブロック内の2つのペンダントブロックを枝で結んだ時に限る．このような場合を避けるため，この2つのペンダントブロックが同じv-ブロックに含まれるような切断点をvとして選べばよい（例えば，図2の切断点a）．なお，図3の切断点aのように，そのa-ブロック内に3つ以上ペンダントブロックが存在する場合には，そのa-ブロック内の2つのペンダントブロックを枝で結んでも，a-ブロック内にペンダントブロックが1つ以上残るためaを切断点とする新しいペンダントブロックは出現しない．

以上の証明に基づくアルゴリズムBICONNECT(G)を次に示す．

```
Procedure BICONNECT(G);
begin
  while Gが非連結 do
    begin
      Gの異なる2つの連結成分を選びG1 , G2 とする;
      if G1 が孤立ブロック
        then v1 :=G1 の任意の点
        else v1 :=G1 に含まれるペンダントブロック中の切断点でない点;
      if G2 が孤立ブロック
        then v2 :=G2 の任意の点
        else v2 :=G2 に含まれるペンダントブロック中の切断点でない点;
      G:=G+{v1 , v2 }; (Gに枝(v1 , v2 )を付加)
    end ;
```

```
Gのd，pを求める；
while max{d-1,⌈p/2⌉}＞2 do
  begin （Gが連結）
    v:=d(v)=dなる切断点；
    if d-1 ＞⌈p/2⌉ then
      begin
        異なる2つのv-ブロックG1，G2 を選ぶ；
        v1 :=G1 に含まれるペンダントブロック中の切断点でない点；
        v2 :=G2 に含まれるペンダントブロック中の切断点でない点；
      end else
      begin (d-1 ≦⌈p/2⌉)
        if (d =2) and (p >3)
          then if v≠v0なる切断点v0があり，しかも2つだけペンダントブロックを含むv0-ブロックにv
                    が含まれる
               then v:=v0；
        ペンダントブロックを2つ以上含むv-ブロックを1つ選びG1 とする；
        G1 以外のv-ブロックを1つ選びG2 とする；
        v1 :=G1 に含まれるペンダントブロックの切断点でない点；
        v2 :=G2 に含まれるペンダントブロックの切断点でない点；
      end;
    G:=G+{v1，v2}；（Gに枝（v1，v2）を付加）
    d，pを更新；（枝付加後のG上でのd，p，qを求める）
  end;
if 1≦i≦2 then
  begin （この場合d＝2，3及びp＝3，4である．）
    p＝4でペンダントブロックを丁度2つ含むv-ブロックがあるときは，まず，そのペンダントブロックの
    1つと，他のv-ブロック中のペンダントブロックとを切断点以外で1本の枝で結ぶ．ペンダントブロック
    （高々4個）がなくなるまで，かってな2つのペンダントブロックを選び切断点以外で1本の枝で結ぶ
  end ；
end ；
```

アルゴリズムBICONNECT に対して次の補題が成立つ．

［補題2］アルゴリズムBICONNECT はGをμ=max{d-1,⌈p/2⌉+q} 本の枝付加で2連結化する．

（証明）補題1の証明より明らか．　（証明終）

3．2　計算時間

ここではアルゴリズムBICONNECT の計算時間がO(|V|+|E|)であることを示す．グラフGが非連結な間（連結になるまで）にかかる計算時間と連結な間（連結になってから2連結になるまで）にかかる計算時間に分けて解析する．

（a）Gが非連結な間にかかる計算時間

Depth First Searchを用いると入力グラフGの各連結成分，ペンダントブロック，孤立ブロックはO(|V|+|E|)の計算時間で求まる．また，異なる2つの連結成分G_1，G_2 に枝を付加して1つの連結成分にする時，G_1，G_2 とも孤立ブロックであるなら枝付加後G_1，G_2 はペンダントブロックになる（図4(a)）．G_1 あるいはG_2 の片方例えばG_1 だけが孤立ブロックであるなら枝付加後G_1 はペンダントブロックとなり，枝を付加されたG_2 のペンダントブロックはペンダントブロックでなくなる（図4(b)）．またG_1，G_2 とも孤立ブロックでないなら，枝で結ばれるG_1，G_2 の2つのペンダントブロックはペンダントブロックでなくなる（図4(c)）．このように枝付加後のGの連結成分，及びそれに含まれるペンダントブロック，孤立ブロックは枝付加前のGから

直接求まる．さらに，枝を１本付加したことによるｐ，ｄ，ｑの値の変更も次のように定数時間でできる．

（Ⅰ）G_1 及びG_2 がともに孤立ブロックの場合

ｐ:＝ｐ＋２，ｑ:＝ｑ－２，ｄ:＝ｄ－１

（Ⅱ）G_1 あるいはG_2 の片方だけが孤立ブロックの場合

ｐ変化なし，ｑ:＝ｑ－１，ｄ:＝ｄ－１

（Ⅲ）G_1 ，G_2 とも孤立ブロックでない場合

ｐ:＝ｐ－２，ｑ変化なし，ｄ:＝ｄ－１

連結成分がｎ個あったとするとｎ－１本の枝付加で連結グラフになり，連結成分やペンダントブロックを見つける前処理の時間を除けば１本当り定数時間で付加できる．よってＧが非連結な間の操作にかかる計算時間は全体でもO(|V|+|E|)である．

（ｂ）Ｇが連結な間の計算時間

Ｇ上で直接v-ブロックやペンダントブロックを探索すると，線形時間でＧを２連結化することは難しい．このため，ＧのＢＣ木T(G)を作成し，T(G)上でv-ブロックやペンダントブロックを探索する．T(G)の枝数，点数は明らかにO(|V|)であり，T(G)の隣接リストＡＬはO(|V|+|E|)の計算時間で作成できる．T(G)の任意のＣ点をｖとし，ｖに対応するＧの切断点をｖ' としよう．また，ｖから任意２つの葉 ℓ_1 ， ℓ_2 への２本の枝素な道をQ_1 ，Q_2 としよう．このとき ℓ_1 ， ℓ_2 に対応するＧの２つのペンダントブロックP_1 ，P_2 は２つのv'- ブロックB_1 ，B_2 にそれぞれ含まれている（図５）．T(G)は木なので，ｖに隣接している異なる２つのＢ点v_1 ，v_2 を選び枝（ｖ，v_1 ），（ｖ，v_2 ）から始めてT(G)上の枝を任意の葉 ℓ_1 ， ℓ_2 に到達するまで探索していけば，ｖから ℓ_1 までの道Q_1 とｖから ℓ_2 までの道Q_2 が定まる．むろんQ_1 とQ_2 は枝素な道である．ＡＬは図６のような双方向リストであり，枝（v_i ，v_j ）に対応する要素がリストAL(v_i ）及びAL(v_j ）の両方に含まれ，互いにポインタで結ばれており，片方から他方が直接アクセスできる．図６の太線で示すようなリストのたどり方をすると，道（v_i ，v_j ），（v_j ，v_k ）を見つけることができる．この隣接リストＡＬを用いるとQ_1 とQ_2 が含む枝の本数の枝探索でQ_1 ，Q_2 が求まり，異なるv-ブロックB_1 とB_2 に含まれるペンダントブロックP_1 とP_2 を見つけたことになる．．

ｄ－１＞⌈ｐ／２⌉またはｐ＝ｄ≦３の場合には上のようにして見つけたペンダントブロックを枝で結べばよい．従って，Q_1 ，Q_2 の長さ（枝の本数）をそれぞれm_1 ，m_2 とすれば，明らかにＯ（m_1 ＋m_2 ）時間で実行できる．

ｄ－１＞⌈ｐ／２⌉またはｐ＝ｄ≦３のどちらでもない場合にはBICONNECT は２つ以上のペンダントブロックを持っているv-ブロックを１つ選ぶ必要がある．つまり，B_1 ，B_2 の少なくともどちらか一方は２つ以上ペンダントブロックを含んでいるかどうか判定しなければならないが，これはQ_1 （Q_2 ）上に次数が３以上のｖ以外の点があるかどうか判定すればよい．なぜなら次数３以上の点はQ_1 （Q_2 ）上の点以外の点と隣接しているため，B_1 （B_2 ）に対応するT(G)の部分木T_1 （T_2 ）には ℓ_1 （ ℓ_2 ）以外の葉が存在することを意味するからである（図５）．このようにしてQ_1 （Q_2 ）を見つける時にQ_1 （Q_2 ）上の各点v_i が３次以上かどうかを調べていけばよい．よって，高々Q_1 とQ_2 が含む枝の本数の３倍の枝探索で異なる２つのv-ブロックに含まれるペンダントブロックを定めることができ，かつそれらのv-ブロックが２つ以上のペンダントブロックを持つかどうか判定できる．

B_1 ，B_2 の少なくともどちらか一方が２つ以上のペンダントブロックを持つ場合には，道Q_1 ，Q_2 を見つけることにより得られた２つのペンダントブロックP_1 ，P_2 を選ぶ．もし両方とも１つずつしかペンダントブロックを含んでいない場合，つまりQ_1 ，Q_2 に含まれるｖ以外の点がすべて次数２以下の場合には，P_1 ，P_2 を選ぶことはできない．この時道Q_1 ，Q_2 をそれぞれ１本の枝（ｖ，ℓ_1 ），（ｖ，ℓ_2 ）に置き換えてから新たに２つの道を探す（図７）．従って最終的に選ばれた道Q_1 ，Q_2 の長さをそれぞれm_1 ，m_2 とし，Q_1 ，Q_2 以外のたどられた枝の本数をm_3 とすると，枝付加できる２つのペンダントブロックがＯ（m_1 ＋m_2 ＋m_3 ）の計算時間で見つかる．

さらに，ｄ＝２かつｐ＞３の場合には，$v_0 \neq$ ｖなる切断点v_0があり，ちょうど２つだけペンダントブロックを含むv_0－ブロックＢにｖが含まれているかどうかを確める必要があり，もしそうならばv_0を新たにｖとして選び直してペンダントブロックの探索を行なっていく．T(G)を用いるとこの操作は次のようになる．

まず，最初に選択したｖ（次数は２であることに注意）から適当にT(G)上をたどって２つの葉 ℓ_1 ， ℓ_2 まで

の道Q_1，Q_2を求める．Q_1，Q_2が次のような形をしているかどうか調べる（図8(b)）．

$Q_1 = (v, v_1), \cdots, (v_n, \ell_1)$

$Q_2 = (v, u_1), \cdots, (u_i, u_{i+1}), \cdots, (u_m, \ell_2)$ （1）

ただし，ℓ_1，ℓ_2は葉，u_iは次数3のB点，u_i以外のQ_1，Q_2上のB点の次数はすべて2であるとする．Q_1とQ_2が(1)の形をしている時にはu_iに隣接する切断点v_0（$\neq v$）があり，ℓ_1，ℓ_2に対応する丁度2つだけのペンダントブロックを含むv_0−ブロックBがありvを含む．このときℓ_1，ℓ_2に対応するGの2つのペンダントブロックに枝付加すると切断点v_0のまわりに新しいペンダントブロックが出現するため，v_0に隣接しているu_i以外の点w_1を選び，道Q_2を

$Q_2 = (v, u_1), \cdots, (u_i, v_0), (v_0, w_1), \cdots, (w_s, \ell_3)$ （2）

のように選び直す（図8(c)）．また$(u_{i+1}, u_{i+2}), \cdots, (u_m, \ell_2)$を1本の枝$(u_{i+1}, \ell_2)$に置き換える．この時選び直した$Q_1$，$Q_2$から得られる葉$\ell_1$，$\ell_3$に対応するGのペンダントブロックは$v_0$の異なる2つの$v_0$−ブロックにそれぞれ含まれている．つまり，$v_0$を切断点として，$v_0$から2本の枝素な道を探索して2つの葉を定めたのと同じことになる．

Q_1，Q_2が(1)のような道であるかどうかの判定はQ_1とQ_2が含む枝数に比例した計算時間でできる．もし，Q_1，Q_2が(1)のような道である時は(2)のように選び直さなければならない．このとき(1)の道に含まれる枝で(2)の道に含まれていない枝$(u_{i+1}, u_{i+2}), \cdots, (u_m, \ell_2)$を1本の枝$(u_{i+1}, \ell_2)$で置換えなければならないがこの置き換えは直ちにできる．最終的に選ばれた道をQ_1，Q_2とし，その長さをそれぞれm_1，m_2とする．またそれら以外のたどられた枝数をm_3とすると，この場合にも枝付加できる2つのペンダントブロックが$O(m_1 + m_2 + m_3)$時間で見つかる．

Gの2つのペンダントブロックP_1，P_2からそれぞれ切断点以外の点を1つずつ選び枝を付加すると，いくつかの2連結成分が1つの2連結成分になり，いくつかの切断点が切断点でなくなる（図9(a)(b)）．枝付加後のグラフをG′とした時，T(G)をT(G′)に変更しなければならない．次のアルゴリズムUPDATEにより効率良くこの変更ができる．ここで，GのBC木上のB点とC点を結ぶ2次の点のみからなる任意の道を1本の枝で置き換えてできる木もBC木と再度定義し直しT(G)で表わすことにする．このことは，その道上の全てのB点に対応する2連結成分を1つの2連結成分として考えることを意味する．

Algorithm UPDATE

ステップ1．Gの切断点でG′において切断点でなくなるのはT(G)上のQ_1，Q_2に含まれる次数2のC点に対応する切断点である（図10(a)）．このため，T(G)からこのような切断点をまず除去する（図10(b)）．

ステップ2．Q_1，Q_2のB点全体に対応するGの2連結成分はG′では1つの2連結成分となるため，Q_1，Q_2に含まれるB点の中から1点v_Bを適当に選び，Q_1，Q_2上の他のB点すべてをv_Bに短絡する（図10(c)）．

ここで次の補題が成立つ．

［補題3］アルゴリズムUPDATEはT(G)をT(G′)に変更する．

（証明）UPDATEを実行した後のBC木をT′とする．枝付加によりG′ではQ_1，Q_2上のB点に対応するGの2連結成分が1つの2連結成分となるが，これはT′のv_Bに対応する．これ以外のG′の2連結成分はGの2連結成分と同じであり，これに対応するTのB点はT′において変更を受けていない．ステップ1において，G′において切断点でなくなるC点はT′から除去され，他のC点はそのままT′上に残る．よってT′はG′のBC木T(G′)である．

（証明終）

データ構造として図6の隣接リストを使い，Q_1，Q_2を見つける時に次数2のC点を記憶しておけば，アルゴリズムUPDATEのステップ1の点除去は1個当り定数時間で実行できる．　また，ステップ2で短絡する時に多重枝が生じるので，AL上でこの多重枝を1本の枝に置換えなければならないが，Q_1，Q_2に含まれるC点を記憶しておけばこれも1本当り定数時間でできる．ステップ2でQ_1，Q_2上にあるB点v′をv_Bに短絡するには，AL(v′)につながっている要素をAL(v_B)に追加するようにつなぎ換えればよい．この追加はQ_1，Q_2を定めるときにたどったAL(v′)とAL(v_B)の要素間のポインタのつなぎ換えにより実行でき，B点の番号v′，v_Bを知る必要はない．図6のデータ構造を用いるとこのつなぎ換えは定数時間でできる．ただし，点v′

の隣接点は短絡によりv_Bに隣接するようになるが，AL（v_B）の各要素の点番号をv'からv_Bにいちいち書き換える必要はない．図6のデータ構造を用いると枝の一方の端点から他方の端点へ直接ポインタをたどっていけるため各要素で点番号を参照することなく葉ℓ_1，ℓ_2までの道Q_1，Q_2を見つけることができる．しかも，Q_1，Q_2上の各点の次数が3以上かどうかは，AL上でQ_1，Q_2を見つける時に探索した各要素のleftとrightのポインタが同一の要素を示しているかどうかで判断できる．よって，点番号を書換えなくともALを探索して異なる2つのv-ブロックに含まれるペンダントブロックを見つけることができ，かつそれらのv-ブロックが2つ以上ペンダントブロックをもつかどうか判定できる．以上からステップ1，ステップ2の処理は1つの点当り定数時間ですむため，アルゴリズムUPDATEは$O(m_1+m_2+m_3)$の計算時間でT(G)をT(G')に変更する．

以上からT(G)をデータ構造として用いると枝付加するペンダントブロックはどの場合も$O(m_1+m_2+m_3)$の計算時間で見つかり，T(G')もT(G)からその計算時間で求まる．ここで，G'はGに枝を1本付加した後のグラフである．

次にd，pの更新が$O(m_1+m_2+m_3)$の計算時間でできることを以下に述べる．

（1）pの変更

枝付加する2つのペンダントブロックを見つけるのに探索した2本の道Q_1，Q_2が両方とも次数2以下の点しか含まない場合，枝付加により新しいペンダントブロックが1つできるため，p:=p−1とし，それ以外はp:=p−2とする．

（2）dの変更

枝付加によりいくつかのC点の次数が変化するが，これに伴うdの変更を$O(m_1+m_2+m_3)$時間で実現するため，各C点の次数を図11で示すようなデータ構造でおさえておく．つまり，リストDA[i]に次数iのC点vをいれた要素をすべてつないでおく．またC点の個数だけのポインタの配列CAを用意し，CA[v]で直接DA[i]中の要素vをアクセスできるようにしておく．その要素に次数iも収めておく．このデータ構造はT(G)から$O(|V|+|E|)$で作成できる．枝付加により次数が変わるC点はQ_1，Q_2上のC点だけであり，それらのC点は丁度1だけ減少する．点vがQ_1またはQ_2上のC点である時，リストDA[i]につながっている要素vをCA[v]から直接アクセスし，その要素をリストDA[i]からリストDA[i-1]につなぎ換え，vの次数としてi−1をその要素に代入すれば，点vの次数変更に伴うリストDAの変更は1個当り定数時間でできる．よって，図11のデータ構造を用いると$O(m_1+m_2)$の計算時間でC点の次数変化に伴うリストDAの変更が実行できる．このような変更をした後，リストDA[d]がNILとなったならばd:=d−1，そうでなければdは変更しない．このようにして$O(m_1+m_2+m_3)$の計算時間で枝付加後のdの値が求まる．また，DA[d]のリストを参照することにより直ちにd(v)=dである点vも見つかる．

以上から次の補題が成立つ．

［補題4］アルゴリズムBICONNECTは$O(|V|+|E|)$の計算時間でGを2連結化する．

（証明）i本目の枝をGに付加するのに，T(G)上で探索された枝の本数を$m_{1i}+m_{2i}+m_{3i}$とする．m_{1i}，m_{2i}，m_{3i}はi本目の枝を付加した時のm_1，m_2，m_3である．このとき，Gを2連結化するのに探索した枝の本数は

$$\sum_{i=1}^{\mu(G)}(m_{1i}+m_{2i}+m_{3i})=\sum_{i=1}^{\mu(G)}(m_{1i}+m_{2i})+\sum_{i=1}^{\mu(G)}m_{3i}$$

となる．ここで$\mu(G)=\max\{d-1,\lceil p/2\rceil\}$であり，i本目の枝付加が道$Q_{1i}$と$Q_{2i}$の葉$\ell_{1i}$と$\ell_{2i}$に対応するペンダントブロック$P_{1i}$と$P_{2i}$の間になされたとしている．アルゴリズムUPDATEでT(G)をT(G')に変更するとき点除去や点短絡により，Q_{1i}とQ_{2i}に含まれる枝の少なくとも半分，すなわち$(m_{1i}+m_{2i})/2$本の枝がT(G)から除去される．従って，Gが2連結グラフになるまでには最初にあったT(G)の枝だけが除去されるので，T(G)の高々2倍の本数の枝をQ_1，Q_2として走査することになる．よって，

$$\sum_{i=1}^{\mu(G)}(m_{1i}+m_{2i})\leq O(|E|)$$

である．従って，

$$\sum_{i=1}^{\mu(G)} m_{3i} \leqq O(|E|)$$

となることを証明すればよい．

そこで，C点vが道探索の始点として選ばれ，探索された道が次数２以下の点しか含まずQ_1，Q_2として採用されなかった場合を考える．今，この道をv，w，‥‥，ℓ（葉）とすると，この道は１本の枝（v，ℓ）に置換えられるため，これ以降この道上の枝は再度探索されることはない．ここで，置換えた枝（v，ℓ）は，vに隣接する他のどの枝（v，w）よりも後に探索されるようにしておく．そのうえで枝（v，ℓ）が探索されたときのことを考える．点vがこの道探索の始点である場合にはC点vから出る全ての道は既に各々１本の枝で置換えられているはずであり枝（v，ℓ）は必ずQ_1かQ_2として採用され，アルゴリズムUPDATEにより除去される．一方v以外の点uが道探索の始点で，かつ枝（v，ℓ）がQ_1またはQ_2に含まれたならアルゴリズムUPDATEによりこの枝は除去される．Q_1とQ_2どちらにも含まれないときは，この時探索された道u，‥‥，v，ℓ上の枝は１本の枝に置換わるため，（v，ℓ）は再度探索されることはなくなる．以上からQ_1，Q_2以外として探索された枝は必ず除去されるのでどの枝も高々１回しか探索されないことが解る．さらに，置換えにより追加される枝の総数は明らかにn本以下であるので，

$$\sum_{i=1}^{\mu(G)} m_{3i} \leqq O(|E|)$$

である．　（証明終）

本章では，文献(1),(2) のアルゴリズムの不備な点を正した線形時間２連結化アルゴリズムBICONNECT を与えた．アルゴリズムBICONNECT はＢＣ木を表現するデータ構造として一般的なデータ構造である隣接リストを用いているため，特殊なデータ構造を用いている文献(2) のアルゴリズムよりリスト処理等インプレメンテーションも直観的にわかり易くなっている．

4　平面グラフ２連結化アルゴリズム

4．1　アルゴリズム

ここでは，平面グラフGに最小本数の枝を付加して２連結平面グラフにする線形時間アルゴリズムPBICONNECT(G) を与える．ただし，Gの平面埋め込みは固定して変更しないとする．平面グラフの２連結化の場合，次の点が一般グラフのアルゴリズムと異なる．

（ａ）枝付加後も平面性を保持しなければならないため，異なる面に含まれるペンダントブロックや孤立ブロックを枝で結ぶことはできない．図12に示す平面グラフの場合，ペンダントブロックＡとＢは同じ面に含まれているため枝付加しても平面性は保持されるが，ＣとＤはそれぞれ異なる面に含まれているため枝付加すると必ず点以外で枝が交差し平面性が損われる．このため，各面ごとにその面に含まれているペンダントブロックや孤立ブロックを処理し，２連結化していかなければならない．

（ｂ）ペンダントブロックや孤立ブロックに含まれる点の中から枝を付加する２点を選ぶ時，必ずその外周上の点（例えば図12の点ｕ）を選択しなければならない．ペンダントブロックまたは孤立ブロックの外周以外の内部の点（例えば図12の点ｖ）を選択し，他のペンダントブロックまたは孤立ブロックに含まれる点との間に枝を１本付加すると，この枝は少なくとも前者のペンダントブロックまたは孤立ブロックの外周の枝と交差するため平面性が損われる．

（ａ），（ｂ）から次の補題が成立つ．

［補題５］Gの各面の面グラフをそれぞれ最小本数の枝付加で平面性を保持したまま２連結化すれば，Gに最小本数の枝を付加した平面２連結グラフが得られる．

（証明）（ａ）より各面に含まれるペンダントブロック及び孤立ブロックはそれらが含まれている面内で処理

されなければならない．よって，各面ごとに平面性を保って最小本数の枝付加で２連結化すればG全体を平面グラフのまま最小本数の枝付加で２連結化したことになる．また，（ｂ）より枝付加する点は必ず各面の面閉路上になければならないため面閉路に含まれる点や枝のみを考えれば良い．よって面グラフを２連結化すればよい．

（証明終）

アルゴリズムPBICONNECTの概略を以下に示す．

Algorithm PBICONNECT(G)

ステップ１．Gのすべての面F_iについてその面グラフG_iを求める．

ステップ２．各G_iを最小本数の枝付加でかつ平面性を保持したまま２連結化する．

ステップ２で各G_iを２連結化するが，この操作に一般グラフの２連結化アルゴリズムBICONNECT を用いる．ただし，平面性を保持するように枝付加しなければならないためG_iが連結の場合には枝付加するペンダントブロックの選び方に少し制約を加える．G_iが連結の場合２つのペンダントブロックP_1，P_2を任意に選択し枝付加すると，P_1，P_2が含まれていた面F_1が２つの面F_2，F_3に分割される．このとき，同じ面F_1に含まれていたP_1，P_2以外のペンダントブロックが異なる２つの面F_2，F_3にそれぞれ分離して含まれることがあり，F_2に含まれるペンダントブロックとF_3に含まれるペンダントブロックの間では枝付加できなくなることがある．図13(a) のグラフの面F_iに着目する．ペンダントブロックAとBの間に枝付加すると図13(b) の点線で示した場合のようにペンダントブロックBとDが異なった面に含まれてしまい，BとDの間には枝を付加できなくなる．アルゴリズムBICONNECT をそのままG_iに適用すると，これら枝付加できない組合せが生じ，かつそれらを枝で結ぶ場合がある．BICONNECT を用いて平面性を保ったままG_iを２連結化するときはP_1，P_2の選び方に制約を加えて枝付加した後も他のペンダントブロックがすべて同一の面内に含まれるようにしながら行なう．このためアルゴリズムBICONNECT に以下のような変更(1)(2)を加える．

（１）BICONNECT では，v-ブロックG_2としてG_1以外のv-ブロックならどれを選んでもよかったが，ここでは切断点ｖのまわりでv-ブロックG_1の隣りに埋め込まれているv-ブロックをG_2として選ぶ（図14）．

（２）G_1，G_2からそれぞれ枝付加するペンダントブロックを選ぶ時，BICONNECT では適当に１個ずつ選択してよかったが，ここでは隣り合ったペンダントブロック同士を選ぶようにする（図14の斜線で示されたペンダントブロック）．

このようにすれば残りのペンダントブロックが２つの面に分離されることはない．ここで次の補題が成立つ．

［補題６］変更(1)(2)を加えたアルゴリズムBICONNECT は連結グラフG_iを平面性を保ったまま最小本数の枝付加で２連結化する．

（証明）変更(1)(2)により枝付加後もG_iの残りのペンダントブロックは１つの面に含まれている．よって，平面性を保ったまま枝付加が行なわれる．また，アルゴリズムBICONNECT では，v-ブロックやペンダントブロックはかってに選択してよかったが，変更(1)(2)はこの選択に制約を加えることになる．この制約のもとでも枝付加できるペンダントブロックが必ず見つけられるので，BICONNECT は$\max\{d_i-1, \lceil p_i/2\rceil\}$本の枝付加で$G_i$を２連結化するといえる．ここで$d_i$，$p_i$は$G_i$におけるp，dである．　（証明終）

G_iが非連結の場合には，アルゴリズムBICONNECT をそのまま用いても平面性は保たれるため，アルゴリズムPBICONNECTはステップ２でG_iが非連結の場合には，アルゴリズムBICONNECT をそのまま適用し，G_iが連結な場合にのみ変更(1)(2)を加えたBICONNECT を適用する．このため，アルゴリズムPBICONNECT(G) は各$G_i \in G$を$\max\{d_i-1, \lceil p_i/2\rceil+q_i\}$本の枝付加で２連結化する．$d_i$，$p_i$，$q_i$は$G_i$のd，p，qである．よって，アルゴリズムPBICONNECT(G) は

$$\sigma(G)=\sum_{G_i\in G}\max\{d_i-1, \lceil p_i/2\rceil+q_i\}$$

本の枝付加でG全体を平面２連結化する．Gを平面２連結化するには少なくとも$\sigma(G)$本の枝付加が必要であることは補題 5,6より明らかであるため以下の定理が成立つ．

［定理１］アルゴリズムPBICONNECTは平面グラフGを平面性を保ったまま最小本数σ(G) の枝付加で２連結化する．（定理終）

４．２　計算時間

ここではアルゴリズムPBICONNECTが線形時間アルゴリズムであることを示す．Gの各面F_i の面グラフG_i はF_i の面閉路を時計回りにたどることによって求まる．図 4.9のようなデータ構造をもつGの平面埋め込みリストを用いると，O(|E|)の計算時間ですべてのG_i が求まる．以下で変更(1)(2)を加えたアルゴリズムBICONNECTが各G_i をO（$|V_i| + |E_i|$）の計算時間で２連結化することを示す．ここで，G_i の枝集合をE_i，点集合をV_i とする．

（ａ）G_i が非連結な場合

アルゴリズムBICONNECT と同じ動作をするのでO（$|V_i| + |E_i|$）の計算時間でG_i を２連結化する．

（ｂ）G_i が連結な場合

G_i の平面埋め込みどうりに埋め込まれたＢＣ木T(G_i) を作成する必要があるが，これはF_i の面閉路を時計回りに一巡することによりO（$|V_i| + |E_i|$）の計算時間でつくれる．このT(G_i) を用いると，BICONNECT に付加した変更(1)(2)は次のような操作に置き換わる．ここで，枝付加する２つのペンダントブロックを見つけるために，T(G_i) 上で求める２本の道を

$Q_1 = (v, v_1), (v_1, v_2), \cdots, (v_n, \ell_1)$　$Q_2 = (v, u_1), (u_1, u_2), \cdots, (u_m, \ell_2)$

とする．ただし，点vはT(G_i) 上のC点，及びℓ_1，ℓ_2 は葉である．

（１）点vの回りで枝（v, v_1）の反時計回りに隣の枝をQ_2 の最初の枝（v, u_1）として選ぶ．

（２）Q_1 を探索する時，枝（v_{i-1}, v_i）の次の枝（v_i, v_{i+1}）は点v_i の回り，枝（v_{i-1}, v_i）の時計回りに隣の枝を常に選ぶ（図15）．Q_2 を探索する時，枝（u_{i-1}, u_i）の次の枝（u_i, u_{i+1}）は点u_i の回り，枝（u_{i-1}, u_i の反時計回りに隣の枝を常に選ぶ（図15）．

変更(1)(2)を付加したアルゴリズムBICONNECT が見つけたP_1，P_2 の葉ℓ_1，ℓ_2 は明らかに隣合ったペンダントブロック同士に対応している．T(G_i) の埋め込み隣接リストとして図６のデータ構造を用いると，時計回りに隣の枝，反時計回りに隣の枝は即座にアクセスできるため，変更(1)(2)をアルゴリズムBICONNECT に付加しても枝探索に要する時間は変化しない．一般グラフに適用するアルゴリズムBICONNECT と各G_i に適用する平面性を保持したアルゴリズムBICONNECT との間の相異点は道Q_1，Q_2 を見つけるための枝探索に，以上の変更(1)(2)を付加するかしないかという点だけなので，(1)(2)を付加した平面グラフ用のアルゴリズムBICONNECT はO（$|V_i| + |E_i|$）の計算時間で各G_i を平面２連結化する．以上から次の補題が成立つ．

［補題７］アルゴリズムPBICONNECT(G) はO(|V|+|E|)の計算時間で平面グラフGを平面性を保ったままで２連結化する．

（証明）Gの各面グラフG_i はO(|E|)で求まり，各G_i はO（$|V_i| + |E_i|$）で平面２連結化される．よって，G全体はO(|V|+|E|)で平面２連結化される．（証明終）

４．３　むすび

本章では，与えられたグラフGの平面埋め込みを固定したままで枝を付加して平面２連結化する線形時間アルゴリズムを与えた．Gの平面埋め込みを固定しない場合には，ペンダントブロックをどの面に埋め込むかで２連結化するのに必要な付加枝の本数，即ちσ(G) が変化する．σ(G) が最小となるようなGの平面埋め込みを求める問題は未解決である．しかし，実用上の問題では，Gの平面埋め込みは固定されており，変更できない場合が多いと思われる．

（参考文献）

［１］K.P.Eswaran and R.E.Tarjan,Augmentation Problems,SIAM J.Comput,5,4,pp653-665,1976

［２］A.Rosenthal and A.Goldner,Smallest augmentations to biconnect a graph,SIAM J.Comput,6,1,pp55-66,1977

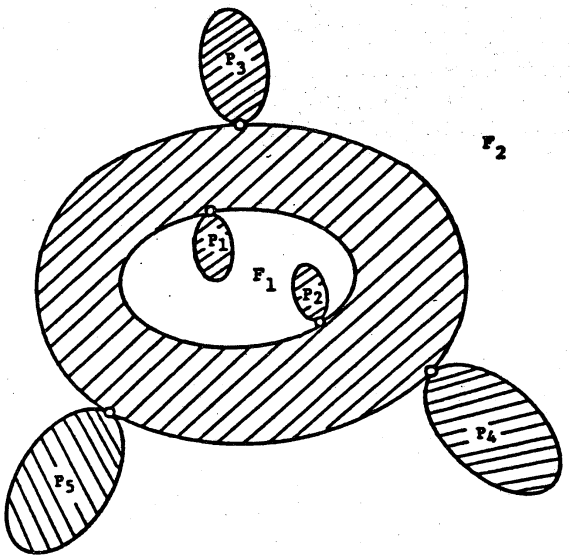


図１　面とペンダントブロックの関係

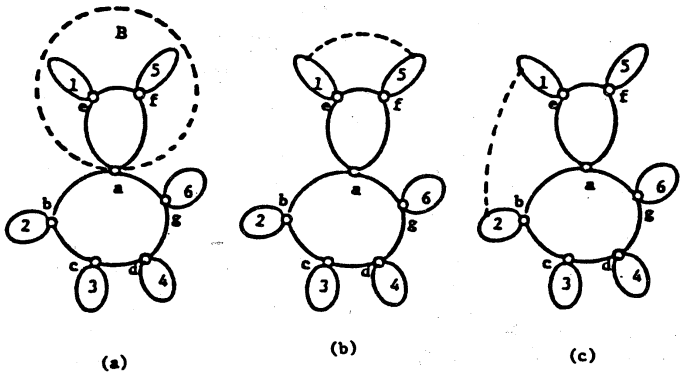


図２　枝付加の特殊例

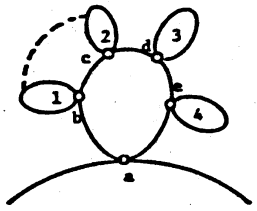


図３　ペンダントブロックができない例

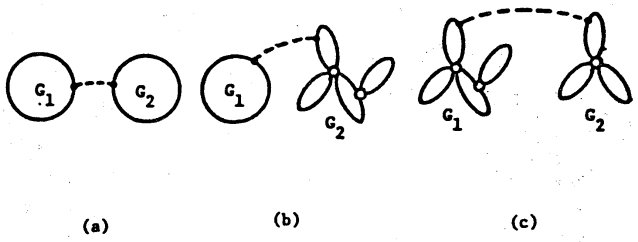


図４　各連結成分間の枝付加

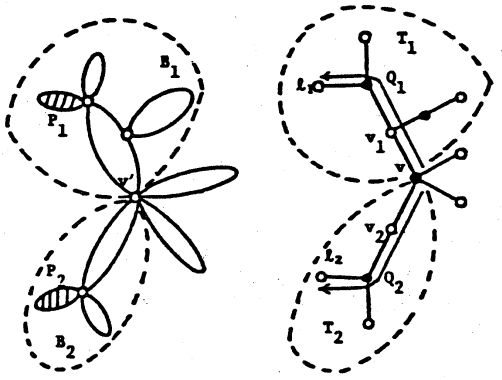


図５　GとT（G）の関係
（a）G　（b）T（G）

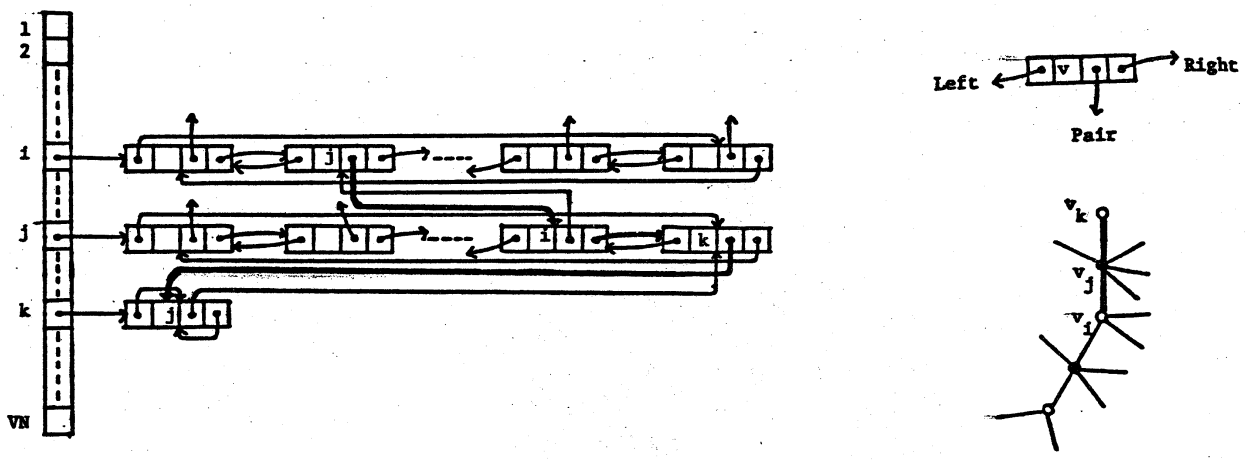


図６　ＡＬのデータ構造

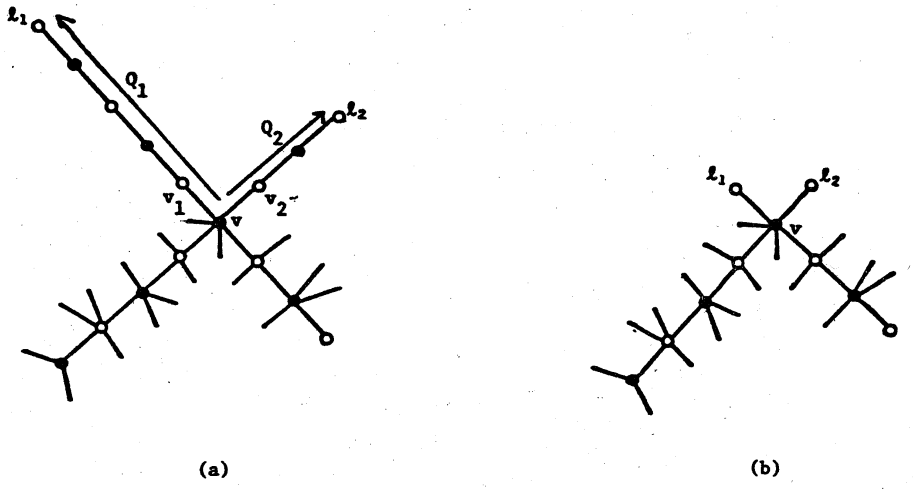


図７　Q_1，Q_2 の短絡
（ａ）短絡前　（ｂ）短絡後

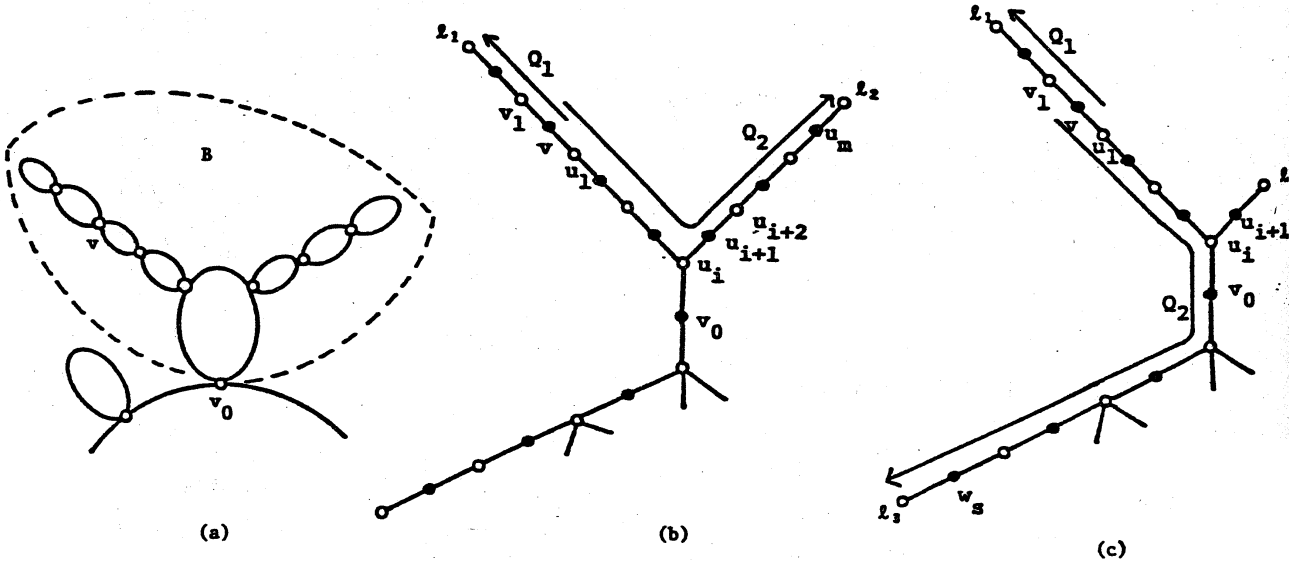


図８　ｄ＝２，かつｐ＞３の場合の処理

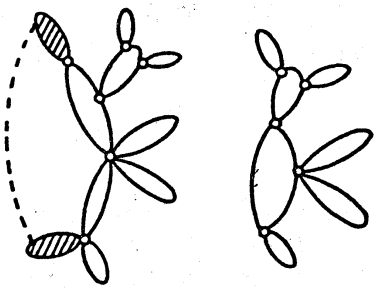
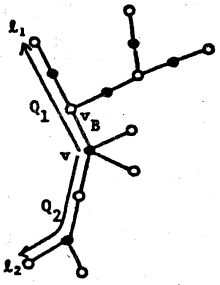


図９　ＧとＧ′
（ａ）Ｇ　　（ｂ）Ｇ′



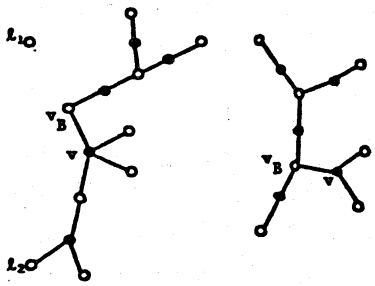


図１０　Ｔ（Ｇ）の更新
（ａ）Ｔ（Ｇ）
（ｂ）次数２のＣ点を除去
（ｃ）Ｔ（Ｇ′）

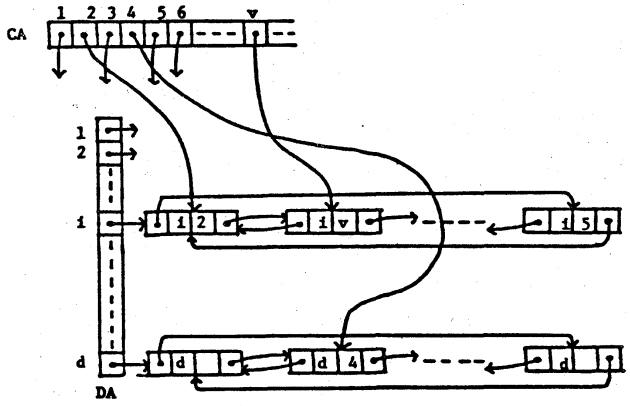


図１１　ＤＡのデータ構造

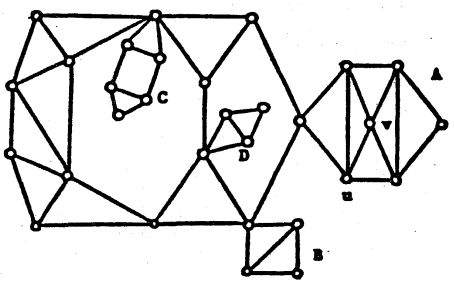


図１２　平面グラフＧ

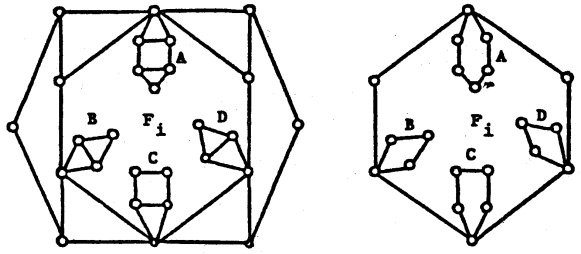


図１３　ペンダントブロックの選び方
（ａ）Ｇ　　（ｂ）G_i

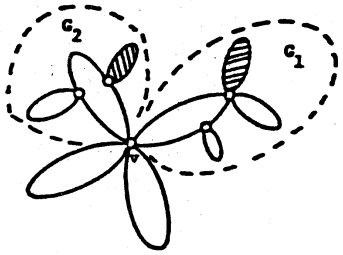


図１４　条件（１）（２）の説明

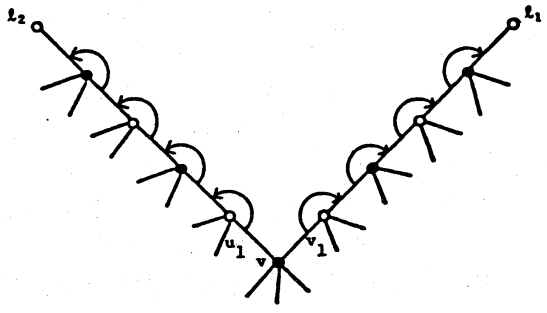


図１５　枝の探索方法